⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平2−224143

⑤Int. Cl.⁵	識別記号	庁内整理番号	㊸公開　平成2年(1990)9月6日
G 06 F 11/28 9/45	340 A	7343−5B	
		8724−5B　G 06 F 9/44　322 Z	

審査請求　未請求　請求項の数 1　(全9頁)

㊹発明の名称　テストプログラム作成処理装置

㉑特　願　平1−46212
㉒出　願　平1(1989)2月27日

㉜発明者　塩谷　睦男　神奈川県川崎市中原区上小田中1015番地　富士通株式会社内
㉛出願人　富士通株式会社　神奈川県川崎市中原区上小田中1015番地
㉔代理人　弁理士　森田　寛　外2名

明　細　書

1．発明の名称

テストプログラム作成処理装置

2．特許請求の範囲

生成規則と意味記述とからなる属性文法記述に従って記述された言語処理プログラムの処理対象文に対して、生成規則に関しての乱数的な選択処理を施すことで該処理対象文の一態様をなす乱数文を生成するとともに、該乱数文の実行結果の確認を行うテストルーチンの呼び出しのための確認文を、該属性文法記述中に該処理対象文と対応付けて記述することで該乱数文と対にして生成して、言語処理プログラム用のテストプログラムとなすテストプログラム作成処理装置であって、

上記属性文法記述を内部形式に変換するリーダ(21)と、

該リーダ(21)により変換された内部形式の内の非終端記号については、生成規則に従って再帰的に展開を実行して対応する内部形式テストプログラムに翻訳し、かつこの展開にあたって必要とされる非終端記号の種々の属性を構造体変数として管理するとともに、終端記号についてはそのまま内部形式テストプログラムとして扱うよう処理する翻訳処理器(22)と、

該翻訳処理器(22)による内部形式テストプログラムの作成にあたって必要となる意味記述の実行を、該翻訳処理器(22)からの起動依頼に応じて実行する意味記述解釈器(27)と、

上記翻訳処理器(22)により作成された内部形式テストプログラムを実行可能なテストプログラムとして出力するライタ(28)とを備えることを、

特徴とするテストプログラム作成処理装置。

3．発明の詳細な説明

(概要)

言語処理プログラム用のテストプログラムを作成するテストプログラム作成処理装置に関し、

テストプログラムの自動作成の提供を目的とし、

属性文法記述に従って記述された処理対象文に対しての乱数的な選択処理により生成される乱数文と、乱数文の実行結果の確認のためのテストルーチンを呼び出す確認文とを対とするテストプログラムを作成するテストプログラム作成処理装置であって、属性文法記述を内部形式に変換するリーダと、変換された内部形式の内の非終端記号については、生成規則に従って再帰的に展開を実行して内部形式テストプログラムに翻訳し、かつこの展開にあたって非終端記号の種々の属性を構造体変数として管理するとともに、終端記号についてはそのまま内部形式テストプログラムとして吸う翻訳処理器と、意味記述の実行を翻訳処理器からの起動依頼に応じて実行する意味記述解釈器と、作成された内部形式テストプログラムを実行可能なテストプログラムとして出力するライタとを備えるよう構成する。

〔産業上の利用分野〕

本発明は、言語処理プログラム用のテストプログラムを作成するテストプログラム作成処理装置に関し、特に、自動的にテストプログラムを作成できるようにするテストプログラム作成処理装置に関するものである。

〔従来の技術〕

従来では、人手によって、言語処理プログラム用のテストプログラムが作成されていた。すなわち、プログラマ等が、各種文法規則とその意味を記述したマニュアルから要因分析を行ってテスト項目を設定して、テスト項目にあった機能の実行とその結果の検証手続きとを記述したテストプログラムを作成していたのである。

〔発明が解決しようとする課題〕

しかしながら、この要因分析やテスト項目の設定には極めて大きな労力がかかることから、従来技術では、テストプログラムの作成に極めて多大な工数がとられてしまうという問題点があった。そして、人手による方法では作成するテストプログラムの内容に限度があることから、従来技術では、言語処理プログラムの機能の検査を十分行っていないという問題点もあったのである。

そこで、本出願人は、先に出願の「特願昭６３－４６５１０号（昭和６３年２月２９日出願、"言語処理プログラム検証方式"）」で、言語処理プログラム用のテストプログラムの自動作成処理装置の発明を案出した。次に、第５図に従って、この特許出願の発明の概要について説明する。

図中、１は構文・意味記述であって、言語処理プログラムの処理対象文を生成規則と意味記述とからなる属性文法記述に従って記述するもの、２はＴＰ自動作成部であって、構文・意味記述１からテストプログラムを自動作成するもの、３はＴＰ自動作成部２が備える生成規則選択部であって、構文・意味記述１中の生成規則に対して乱数的な選択処理を施すことで処理対象文の一態様をなす乱数文を作成するもの、４はＴＰ自動作成部２が備える確認文挿入部であって、作成された乱数文の実行により引き起こされる実行結果の確認を行うためのテストルーチンを呼び出す確認文を乱数文に対応付けて挿入するもの、５はＴＰ自動作成部２が備える記号表であって、乱数文名等を格納するもの、６はＴＰ自動作成部２により作成される一連の検証プログラムであって、言語処理プロ

グラムのテストプログラムをなすもの、7は自動判定部であって、検証プログラム6の乱数文を言語処理プログラムに実行させたときの実行結果が正しいものとなるのか否かを、その乱数文に対応付けられている確認文により呼び出されるテストルーチンに従って自動的に判定するもの、8は結果一覧表であって、自動判定部7の判定結果をプリントアウトするものである。

このように構成される先の特許出願の発明にあっては、プログラマ等により第6図に示すような構文・意味記述1が記述されると、生成規則選択部3は、構文・意味記述1中の処理対象文の生成規則に対して乱数的な選択処理を施すことで処理対象文の一態様をなす乱数文を作成し、確認文挿入部4は、乱数文の正しい実行結果を引数とする確認文を挿入していくことで、言語処理プログラム用のテストプログラムを自動生成していくよう処理することになる。

このような構成をとる先の特許出願の発明の実用性を高めていくためには、第6図に示すような、特定の言語で記述される分かりにくい属性文法記述ではなくて、人間にとって分かり易い属性文法記述から直ちにテストプログラムを作成できるようにする手段を講じていく必要があるとともに、機能追加があるときにも簡単に対応できるようにする手段を講じていく必要があり、更に、作成されていく一連のテストプログラムの実行順序を自由に変更できるようにする手段を講じていく必要があるのである。

本発明はかかる事情に鑑みてなされたものであって、先の特許出願のテストプログラム作成処理装置の発明を一歩進めて、実用性の高いテストプログラムの自動作成処理装置の提供を目的とするものである。

〔課題を解決するための手段〕

第1図は本発明の原理構成図である。

図中、10は属性文法記述であって、検査対象となる言語処理プログラムの処理対象文についての属性文法を記述するとともに、この処理対象文の実行結果の確認を行うテストルーチンの呼び出しのための確認文を記述するもの、11は属性文法記述10をなす生成規則であって、処理対象文の生成規則を記述するもの、12は生成規則11中の終端記号であって、それ以上展開されることのない記号であるもの、13は生成規則11中の非終端記号であって、生成規則11に従って展開されていく記号であるもの、14は属性文法記述10をなす意味記述であって、処理対象文中の意味内容を記述するもの、20は本発明を具備するテストプログラム作成処理装置であって、言語処理プログラム用のテストプログラムを自動作成するもの、21はリーダであって、属性文法記述10を内部形式に変換するもの、22は翻訳処理器であって、リーダ21により変換された内部形式を翻訳して内部形式テストプログラムを作成する展開処理部23と、内部形式テストプログラムの作成にあたって必要となる意味記述14の実行依頼を行う意味記述処理依頼部26とを備えるもの、24は展開処理部23が備える乱数発生部であって、生成規則に関しての選択処理に必要となる乱数を発生させるもの、25は展開処理部23が備える構造体変数管理部であって、内部形式テストプログラムの作成にあたって必要とされる種々の属性を構造体変数として管理するもの、27は意味記述解釈器であって、意味記述処理依頼部26からの起動依頼に応じて処理要求のある意味記述14を実行するもの、28はライタであって、翻訳処理器22により作成された内部形式テストプログラムを実行可能なテストプログラムとしてファイル等に出力するもの、30は出力されたテストプログラムである。

〔作用〕

本発明では、リーダ21の処理に従って、属性文法記述10を翻訳処理器22が解釈し易い形式である内部形式に変換する。この内部形式を受け取ると、展開処理部23は、非終端記号13については、生成規則11に従って再帰的に展開を実行していくことで、内部形式テストプログラムに

翻訳していくとともに、終端記号１２については、そのまま内部形式テストプログラムとして扱うよう処理していく。この内部形式テストプログラムへの翻訳時に意味記述１４の実行が必要となる場合には、意味記述処理依頼部２６は、意味記述解釈器２７を起動していくよう処理し、このようにして起動される意味記述解釈器２７は、構造体変数管理部２５の属性を新たなものに更新していく処理を実行する。そして、展開処理部２３は、内部形式テストプログラムへの翻訳を、この更新された属性を参照しながら実行していくとともに、翻訳にあたって必要となる生成規則１１中に含まれる選択性の規則に関しては、乱数発生部２４が発生する乱数値に従って、選択対象の選択や値の選択を実行していくことになる。

このようにして、内部形式テストプログラムへの翻訳処理が終了すると、属性文法記述１０に記述されている確認文中のパラメータ部分に、処理対象文の一態様として翻訳された乱数文の実行結果に関しての値が設定され、言語処理プログラム用のテストプログラムの基本単位に相当する内部形式プログラムの作成が終了することになる。内部形式プログラムの作成が終了すると、ライタ２８は、内部形式テストプログラムを実行可能なテストプログラム３０として出力して処理を終了する。

以上のように、本発明では、リーダ２１、翻訳処理器２２、意味記述解釈器２７及びライタ２８という構成に従って言語処理プログラム用のテストプログラムの自動作成を実現した。このようにリーダ２１を備えるようにしたことから、属性文法記述１０の記述を特定の言語で記述しなければならないということがなくなり、分かり易い属性文法記述１０に従ってテストプログラム３０を作成できるようになる。そして、意味記述解釈器２７を翻訳処理器２２とは別に備えるようにしたことから、機能追加の要求に対しても意味記述解釈器２７だけの簡単な変更で対応できるようになる。しかも、ライタ２８を備えるようにしたことから、一連のテストプログラム３０の実行順序の変更等が容易に実行できるようになる。

〔実施例〕

以下、実施例に従って本発明を詳細に説明する。

第２図に、本発明により自動作成されるテストプログラムの使用方法の説明図を示す。図中、第１図で説明したものと同じものについては、同一の記号で示してある。４０は検査対象となる言語処理プログラムであって、属性文法記述１０に記述されることになる処理対象文のコンパイル処理を実行するもの、４１はテスト用オブジェクトプログラムであって、言語処理プログラム４０によりコンパイルされたテストプログラム３０のオブジェクトプログラムであるもの、４２はリンカであって、テスト用オブジェクトプログラム４１とライブラリ４３のプログラムとのリンクを実行するもの、４４はリンカ４２によってテスト用オブジェクトプログラム４１とリンクされることになるＣＨＥＣＫプログラムであって、テストプログラム３０中に記述される乱数文の実行結果の確認を行うテストルーチンであるもの、４５は実行単位をなすロードモジュール、４６はロードモジュール４５の実行結果を表す実行結果出力である。

この図に示すように、本発明のテストプログラム作成処理装置２０は、言語処理プログラム４０の機能の検証のために必要となるテストプログラム３０を作成するために用意されるものであり、この必要とされるテストプログラム３０を、マニュアルに従って記述される属性文法記述１０から自動作成するよう処理することになる。

第３図に、「代入文」に関しての属性文法記述１０の一実施例を示すとともに、第４図に、本発明のテストプログラム作成処理装置２０に従って、この「代入文」に関しての属性文法記述１０から自動作成されることになるテストプログラム３０の一実施例を示す。この第３図の属性文法記述１０は、様々な形式の「代入文」を複数個作成して、言語処理プログラム４０がそれらの「代入文」についての計算処理を行ったときに、正しい数値を算出するのか否かをテストするためにプログラマ

等により作成されることになる。この属性文法記述１０と第６図のそれとを比較すれば明らかなように、本発明の属性文法記述１０は、人間にとって極めて分かり易い記述形式となっている。

この第３図の属性文法記述１０から作成されることになるテストプログラム３０は、「代入文」の一態様である乱数文（「代入文」の生成規則に対しての乱数的な選択処理により生成される）と、ＣＨＥＣＫプログラム４４を呼び出すための確認文との対を基本単位にして、この基本単位を複数つなげることで構成される。具体的に説明するならば、第４図中の

ＡＡＥ＝－２９９＊４４８

ＣＡＬＬ　ＣＨＥＣＫ(１,ＡＡＥ,-133952)

がテストプログラム３０の基本単位をなすものである。この基本単位の意味するところは、

ＡＡＥ＝－２９９＊４４８

という乱数文を言語処理プログラム４０で処理させると、変数名の「ＡＡＥ」のところに、

－１３３９５２(＝-299*448)

の値がセットされていれば正しい処理がなされたことになるので、この変数名と値とを引数にして、ＣＨＥＣＫプログラム４４の呼び出しを行う

ＣＡＬＬ　ＣＨＥＣＫ(１,ＡＡＥ,-133952)

という確認文を、この乱数文と対にして記述するのである。ここで、確認文中の引数の"１"は確認文の通番を表している。

従来では、このようなテストプログラム３０は、すべて人手によって作成されていた。これに対して、本発明では、第３図に示すような極めて分かり易い属性文法記述１０から、このような人手により作成されていたテストプログラム３０を自動作成するよう処理することになる。この属性文法記述１０が特定の言語（例えばＬＩＳＰ）でなくて、このような誰にでも分かる記述で構成できるようになったのは、第１図でも説明したように、リーダ２１を備えるようにしたからである。

次に、第３図の属性文法記述１０を例にとりながら、本発明のテストプログラム３０の作成処理について詳細に説明する。

リーダ２１（第１図で説明したものである）は、第３図の属性文法記述１０を読み取ると、この読み取った属性文法記述１０を翻訳処理器２２（第１図で説明したものである）が処理し易い形式である内部形式に変換する。このようにして、内部形式の形で属性文法記述１０を受け取ると、翻訳処理器２２は、内部形式の記号が意味するところの内容に従って、以下のような処理を実行することで、第４図のテストプログラム３０を自動作成していくことになる。

すなわち、出発記号の「¥文法記述」に続いて記述される

（設定　項番値　１）

に従って、システム変数表の「項番値」という変数名のところに、"１"をセットする。この設定処理は、翻訳処理器２２が「設定」という記号を解釈したときに、意味記述解釈器２７（第１図で説明したものである）が管理する「設定」という名の関数に起動をかけることで実行されることになる。この意味記述解釈器２７は、具体的には、例えばＬＩＳＰシステムで構成されて、定義されている関数の呼び出しがかかると、その関数の処理を実行して処理値を翻訳処理器２２に返すよう処理することになる。また、「項番値」というシステム変数は、具体的には、作成されるテストプログラム３０中の確認文の通番を管理する。

翻訳処理器２２は、続いて記述される

＜"ＣＲＬＦ"代入文　５　１０＞　＆

により、所定の数（この例では、５から10までの間の乱数値で指定される）の「代入文」についてのテストプログラム３０を作成する必要があることを知る。ここで、「"ＣＲＬＦ"」は、作成する代入文のテキストを１つずつ改行していけという指定を表す記号であり、また、「＆」は、文末を意味する記号である。この「代入文」は「"」で区切られていないので、そのままテキストとできない非終端記号である。このような非終端記号は、別の位置にどのようにテキストに展開していけばよいかが記述されているので、翻訳処理器２２は、その展開規則（生成規則）に従って展開処理を実

行する。なお、「＊」で区切られた終端記号については、展開処理が実行されることはない。

この例では、続いて、

変数 "－" 式

と「代入文」の定義がされており、更に、この定義中の「変数」がいかなるものであるかが、図中の①部分で展開されており、この定義中の「式」がいかなるものであるかが、図中の②部分で展開されている。この①及び②部分の「｜」の記号は、いずれか1つのものを乱数に従って選択することを表している。これから、翻訳処理器22は、乱数処理に従って「代入文」の「変数」として、例えば「AAE」の変数名を選択し、「代入文」の「式」として、例えば「数1＊数2」の式を選択する。このとき実行する乱数による選択対象の選択は、具体的には、選択対象の前に記述されている数値（この例では、すべて"100"である）の総和の内にある数値を乱数に従って発生させて、その発生した数値がどのゾーンの選択対象に属しているのかで行うことになる。また、「数1＊数2」のように、更に展開されるものについては、その展開規則に従って展開を続行していく。この例では、「数1＊数2」は、更に③部分で展開されているので、その展開に従って、例えば、「数値1」として"－299"を選択し、「数値2」として"448"を選択することになる。

このようにして、第4図のテストプログラム30の先頭に位置する代入文の一態様である

AAE＝－299＊448

というテキストが作成されることになる。なお、②部分の展開規則中に記述されている

（NEQ＊GNTERMV＊0）

は、除算処理の分母の値が"0"でないことが条件になることを表している。

「代入文」の定義に続いて記述される

『代入 変数 （値 式）』

により、翻訳処理器22は、意味記述解釈器27が管理する「代入」という名の関数に起動をかけることで、テスト対象言語用の変数表の「変数」のところに、「式」の値を設定するよう処理する。

いまの場合、「変数」としては「AAE」が選択されているので、この「AAE」という変数名のところに「式」の値を設定する。この「式」の値は、具体的には、②部分の選択規則の「数1＊数2」の後に続いて記述されている

『値設定（＊（値 数1）（値 数2））』

で算出される。この算出処理も、意味記述解釈器27により実行されて、いまの場合では、「数値1」の値である"－299"と、「数値2」の値である"448"との乗算値である"-133952"が、この「式」の値として算出される。これから、「AAE」という変数名のところに、"-133952"が設定されることになる。

「代入文」の定義の最後では、「確認」を展開することが要求されている。この「確認」がいかなるものであるかは、図中の⑤部分で展開されている。これから、翻訳処理器22は、この⑤部分の展開規則に従って、「項番」の値の"1"と、「変数」の「AAE」と、「変数値」の"-133952"とを引数として、CHECKプログラム44の呼び出しを行う

CALL CHECK(1,AAE,-133952)

という確認文の展開を実行する。このようにして、

AAE＝－299＊448

CALL CHECK(1,AAE,-133952)

という内部形式テストプログラムの作成が終了することになる。そして、最後に、⑥部分の更新規則に従って、意味記述解釈器27に起動をかけて、システム変数表の「項番値」の変数名の値を1つ歩進させる処理を行うよう処理する。

このようにして、翻訳処理器22は、以下同様の処理を繰り返していくことで、第4図に示したような内部形式テストプログラムの作成を実行することになる。すなわち、乱数に従って、代入文の様々な態様を実現して、内部形式テストプログラムの自動作成を実行するのである。そして、属性文法記述10に対応する内部形式テストプログラムの作成が終了すると、ライタ28（第1図で説明したものである）は、この作成した内部形式テストプログラムを実行可能なテストプログラム

として出力してテストプログラム３０の作成処理を終了する。

以上の具体例で説明したように、本発明では、翻訳処理器２２は、属性文法記述１０中の非終端記号については、展開規則（主に、終端記号と非終端記号と意味記述とからなる）に従って再帰的に内部形式非終端記号に翻訳していくとともに、属性文法記述１０中の終端記号については、殆ど翻訳せずにそのままのものとして解釈し、属性文法記述１０中の意味記述（非終端記号を展開していくための関数の起動指示や、非終端記号を扱う関数の起動指示等）については、意味記述解釈器２７に起動を依頼していくよう処理する。そして、内部形式の非終端記号を種々の属性からなる構造体変数として扱うよう処理することで、この展開処理を実現することになる。この構造体変数として扱われる属性としては、例えば、展開結果（展開規則によりどのように展開したかを内部形式終端記号、内部形式非終端記号の順序集合体として保持）、値（式の計算結果や変数名など非終端記号を展開した結果の意味的値を保持）、変数表（テスト対象言語用の変数情報やテストプログラム作成上必要となる変数情報を保持）等がある。

〔発明の効果〕

このように、本発明によれば、直観的に分かり易い属性文法記述から、応用範囲の広い言語処理プログラム用のテストプログラムが効率的に自動作成できるようになる。しかも、機能追加は主に意味記述解釈器２７に対して行えばよいので、柔軟性にも富むことになる。そして、リーダ２１が付加されていることから、テストプログラムの生成規則の機能拡張や変更に対しても柔軟に対応できるとともに、ライタ２８が付加されていることから、テストプログラムの作成順番と出力順番を変更したりするといったような加工を施すこともできることになる。これから、本発明を用いることで、テスト効率が著しく高められるとともに、テストの質も著しく高められることになる。

4．図面の簡単な説明

第１図は本発明の原理構成図、

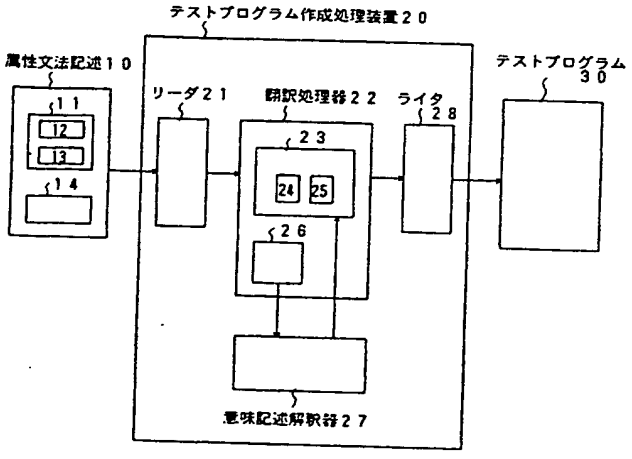
第２図は本発明により自動作成されるテストプログラムの使用方法を説明する説明図、

第３図は属性文法記述の一実施例図、

第４図は第３図の属性文法記述から作成されるテストプログラムの説明図、

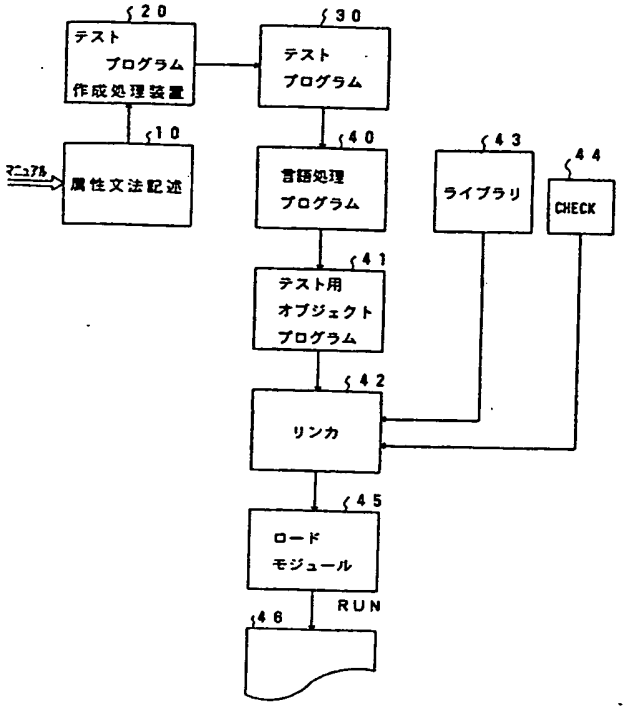
第５図及び第６図は先に出願の発明の説明図である；

図中、１０は属性文法記述、１１は生成規則、１４は意味記述、２０はテストプログラム作成処理装置、２１はリーダ、２２は翻訳処理器、２３は展開処理部、２４は乱数発生部、２５は構造体変数管理部、２６は意味記述処理依頼部、２７は意味記述解釈器、２８はライタ、３０はテストプログラムである。



本発明の原理構成図
第１図

特許出願人　富士通株式会社
代理人　弁理士　森田　寛（外２名）



本発明により自動作成されるテストプログラムの使用方法を説明する説明図

第 2 図

```
[AAE =-299*448
CALL CHECK(1, AAE ,-133952)
[AAA =170
CALL CHECK(2, AAA ,170)
[AAE =-142*-397
CALL CHECK(3, AAE ,56374)
[AAC =495
CALL CHECK(4, AAC ,495)
[AAB =-422+4
CALL CHECK(5, AAB ,-418)
[AAC =-237
CALL CHECK(6, AAC ,-237)
[AAB =301-394
CALL CHECK(7, AAB ,-93)
```

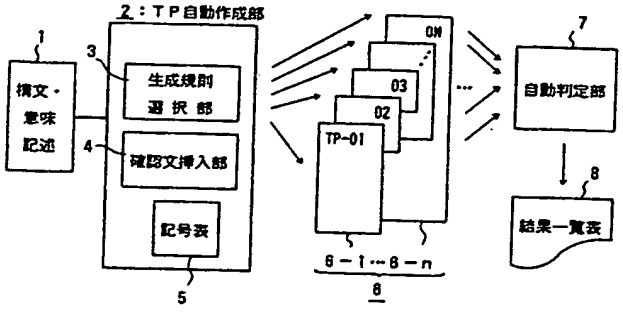
第3図の属性文法記述から作成されるテストプログラムの説明図

第 4 図

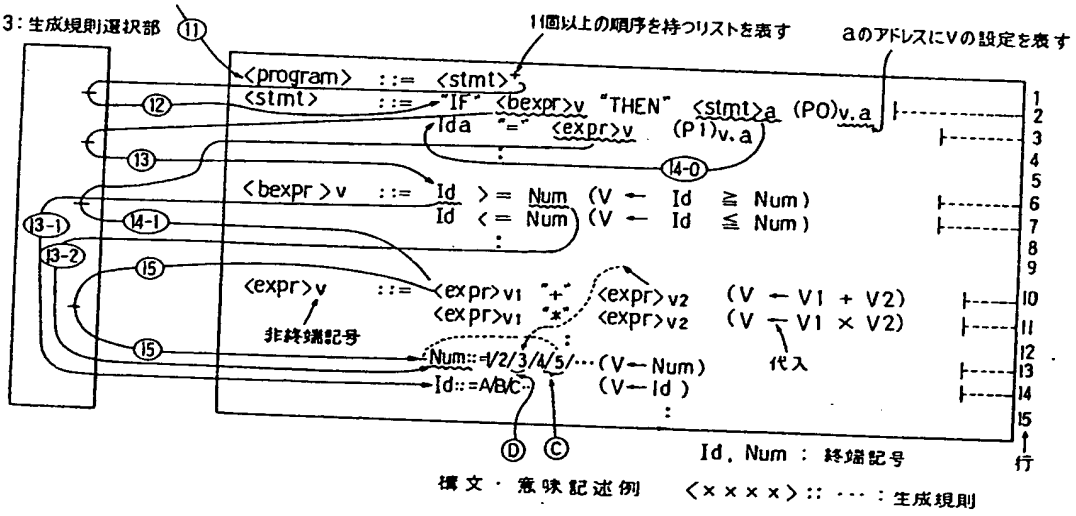
```
＊文法記述 → (設定 項番値 1) <" @NL" 代入文 5 10 > &

代入文 → 変数 " =" 式 " @NL"
          『 代入 変数 (値 式) 』
            確認&

⑤ { 確認 →
    " CALL CHECK( " 項番" , " '変数" , " 「(変数値 変換) 」" ) " &

⑥ {項番 → 「(参照 項番値) 」(設定 項番値 (1+ (参照 項番値))) &

① { 変数 → 100 「' AAA 」 |
          100 「' AAB 」 |
          100 「' AAC 」 |
          100 「' AAD 」 |
          100 「' AAE 」 &

② { 式 → 100 数 |
          100 数1 " + " 数2 『値設定 (+ (値 数1) (値 数2))』 |
          100 数1 " - " 数2 『値設定 (- (値 数1) (値 数2))』 |
          100 数1 " * " 数2 『値設定 (* (値 数1) (値 数2))』 |
          100 数1 " / " ~数2 (NEQ *GNTERMV* 0)
                  『値設定 (/ (値 数1) (値 数2))』 &

③ { 数1 → 数&
    数2 → 数&
    数 → 「(RANDOM -500 500)」 &
```

属性文法記述の一実施例図

第 3 図



先に出願の発明の説明図（I）
第 5 図



先に出願の発明の説明図（II）
第 6 図